12 vs. 16 vs. 18-Bit Computer Horsepower

Noel Lyons and Robert Skyles,
VIDAR Corporation

OBJECTIVE

This paper discusses the differences between 12, 16, and 18-bit word-lengths used in small computers applied to real-time data-acquisition problems. It is our contention that *no particular word length has any substantial advantage* as applied to the type of systems that we make. It is still necessary to select the computer that offers the best performance for the least money, taking into account both hardware and software. In the case of the VIDAR 5206, this turns out to be a 12-bit computer, the PDP-8/L and PDP-8/I! It will be shown that when you investigate computer operations in data acquisition applications, there is no practical, substantial or reasonable advantage to spend extra money for a 16 or 18-bit word-length.

It should be noted that in straight data-acquisition applications, the main limitations to computer system operations are imposed by the IDVM/Scanner. For this discussion, however, we will ignore these and concentrate strictly on 12 vs. 16 vs. 18-bit computer "horsepower."

EVALUATION CRITERIA

Three general criteria are used in discussing the difference between the three different word length computers: speed, memory size and ease of use. We consider speed in reference to the basic computer cycle time. Speed is also related to the number of operations a computer must perform to accomplish a given sub-task. We also consider the size of the computer memory required to do a given job. Third, and probably most important, we present a discussion of 12 vs. 16 vs. 18-bit factors as they apply to the ease of use of the computer/D-DAS system.

COMPUTER OPERATIONS TO BE CONSIDERED

This paper concerns itself with the two major functions of a real-time small computer. First, we judge the instruction set and actual operating instruction sequencing of the computer. We consider the *memory addressing* of the computer and the relationship of this to the 12 vs. 16 vs. 18-bit instruction words and the previously mentioned evaluation criteria. We look specifically at the *memory-reference instructions,* the method of memory addressing with the memory reference instructions, and the register-to-register or *"microinstructions."* The second major computer functions we explore are the various *arithmetic and data storage methods* available with 12, 16 or 18-bit computers. We consider the *arithmetic operations* available, the size of the number systems available, and how the differences affect the evaluation criteria. In addition, we discuss the very important area of *computer data storage,* which is directly related to the arithmetic operating system chosen by the computer user.

COMPUTER MEMORY ADDRESSING

In a computer, memory is addressed by a basic instruction known as a "memory-reference instruction." In the three typical computers we have selected, each of these instructions consists of one 12, 16, or 18-bit word. Figure 1 describes the memory-reference instruction words, which are broken into two parts. The first three or four bits of the word describe the operation to be accomplished and remaining bits describe the memory address that this operation refers to. It is usually implied that with the larger number of bits in the word such as 16 or 18, there is more direct capability of going farther within the core memory to get information. In our sample 12-bit computer, *three* of the 12 available memory-reference instruction bits are used for the operation code, *one bit* is used for a direct/indirect modifier and eight bits are available for memory address. Thus, the 12-bit computer can directly address 2^8 (256) memory locations. Direct addressing implies the use of a single memory-instruction location. In an 18-bit computer, *four* bits are used for the operation code, *one bit* is used for the direct/indirect modifier and 13 bits are available for memory address. There-

fore, with this computer it is possible to directly address 8,192 (2^{13}) memory locations. In the 16-bit computer chosen for this example, once again *four bits* are used for the operation code, *one bit* is used for the direct/indirect modifier, leaving 11 bits available for direct memory address. Therefore, 2,048 memory locations are directly addressable. *It should be noted that many of the "16-bit" machines currently offered are really 8-bit machines and there are no directly addressable memory locations beyond the next immediate location* from which the memory-reference instruction must get its address. In these machines, by strict definition, all memory addressing is done indirectly.

In terms of the directly addressable memory locations, it should be noted that, in the 12 and 16-bit examples used here, only one-half of the directly addressable memory locations are flexibly available to the programmer. The remaining directly addressable memory locations are the "page 0" locations, a set of fixed locations at one end of the computer core memory. This does not apply to the 18-bit machine, in which the user can directly address any location in memory.

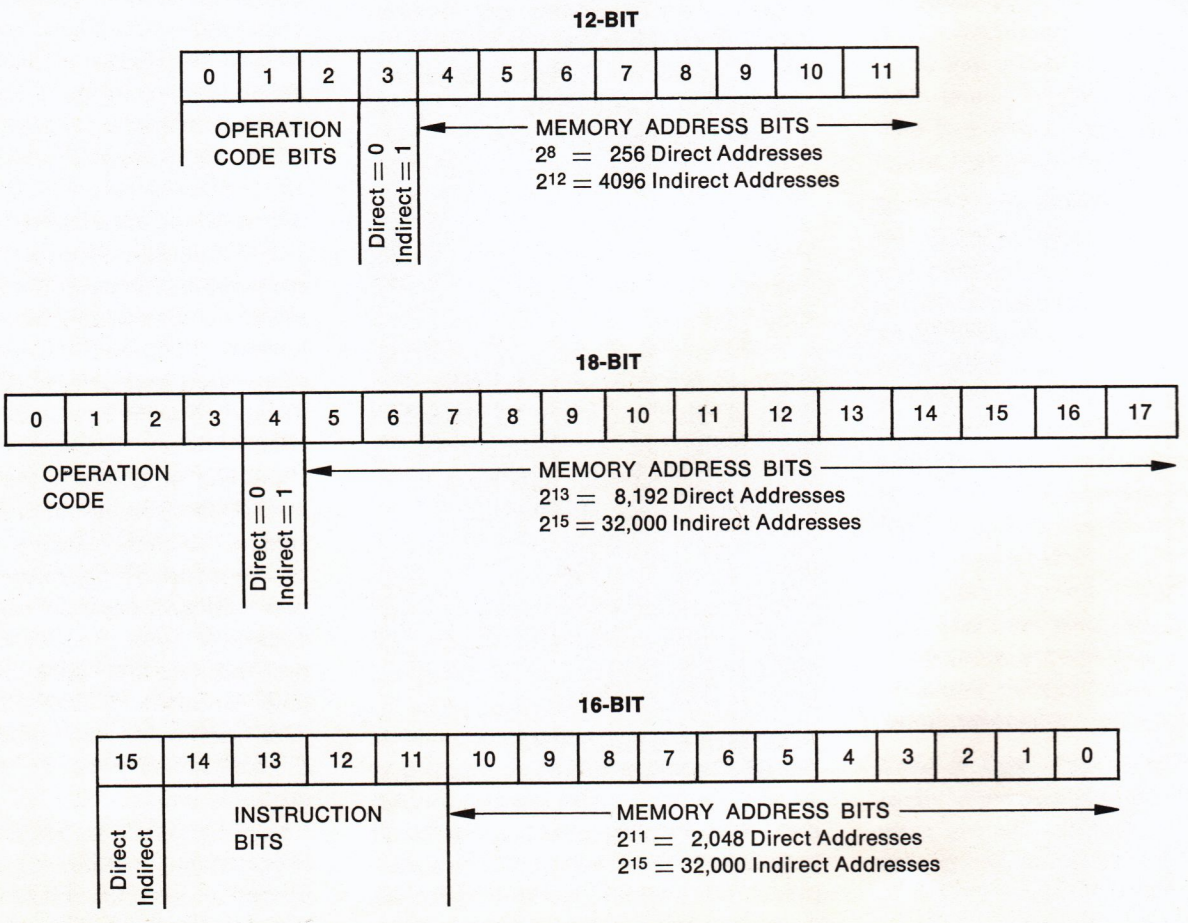
All of the memory locations within a small computer are addressed with a scheme known as *indirect addressing.* In this scheme, the memory-reference instruction word designates a memory location which *contains* the address of the memory location actually desired. Therefore, the 12-bit computer can indirectly address 4,096 memory locations. In 16/18-bit machines, 32,768 memory locations are available through indirect addressing.

Evaluation of Addressing Methods

Speed: Referring back to our three criteria, what are the real differences between these numbers of directly and indirectly addressable memory locations? The major implied difference is that the larger the number of directly addressable memory locations, the faster the speed of operation of the computer. This argument is based on the fact that it takes twice as long to complete an indirectly addressed mem-

Figure 1

Formats of Memory Reference Instructions



ory-reference instruction. Considering that over one-half of the instructions in a typical computer program are basic memory-reference instructions, it would seem that substantial speed improvement could be gained from the ability to directly address large segments of the memory. In actual operation with data-acquisition real-time programs, this has *not proved to be true.* On programs VIDAR has written for data acquisition using the 12-bit PDP-8, counts have been made of the number of times that an indirect memory-reference instruction has been used to arrive at another core location. (It should be noted that there are many occasions when a computer program uses indirect addressing for reasons other than to arrive at a far afield core memory location.) In studying our data-acquisition real-time-program applications, we found that direct-addressing limitations have required using indirect addressing to arrive at far-afield memory locations in about 10 percent of the memory-reference instructions. It should be noted *these were the worst-case examples.* Therefore, in a worst case, you could conclude that *a 12-bit computer is approximately 5% slower in real-time data-acquisition programs than a 16/18-bit computer. We do not consider 5% to be a significant speed difference.* Of course, we have also assumed in these calculations that we were computer limited and not limited by I/O or peripheral-device speeds. Actually, in almost all real-time data-acquisition situations, speed is limited by the I/O devices . . . not the computer program speed. Thus, in practice, *even this 5% difference is only theoretical.*

Memory Requirements: Applying the second evaluation criteria to the direct/indirect memory-addressing problems, it is evident that indirect memory-addressing instructions require one extra memory location apiece. Once again, using real-time D-DAS programs, *approximately 5% extra memory locations are required in a worst-case example* when a 12-bit machine is used instead of a 16-bit machine.

Ease of Use: The third evaluation of 12 vs. 16 bits and direct vs. indirect involves ease of usage. Programming a computer with only 128 directly addressable memory locations immediately adjacent to the current program location involves a lot of bookkeeping. These operations are necessary to keep track of the locations of the various indirect memory addresses. Therefore, a programmer could find the 12-bit machine difficult to use if he had to program in the manufacturer's assembly language. If a relocatable assembler is supplied, this is not true. (VIDAC*, developed for and supplied with the PDP-8 family in the VIDAR 5206, is a relocatable assembler.) In using such an assembler, a programmer is completely unaware of the fact that he has only 256 directly addressable memory locations. He programs the machine as if he had 4,096 directly addressable locations. *All bookkeeping problems are eliminated* because of the automatic-paging assembler and relocatable linking loader. The automatic-paging assembler does the bookkeeping for him. In this case, the user has fewer problems than he would with a 16-bit machine where he has only 2,048 directly addressable memory locations and had to do bookkeeping to keep track of the other 2,048 indirectly addressable locations.

Another factor to be considered is the way in which typical real-time programs are written for data acquisition. The common method of writing a real-time D-DAS program is to write the program to fit in the bottom half of the core memory and to leave the upper half open for storage. In most systems, the program is separated by 2,000 to 3,000 memory locations from the data storage area. In these cases, the 1,024 directly addressable adjacent locations of the 16-bit computer do not offer any useful advantage. In ease of usage, then, *software, and not word length is most important.*

MEMORY REFERENCE INSTRUCTION AVAILABILITY

The second major topic concerns the basic operations performed by the computer. How are these operations affected by the number of bits in a computer word? The majority of a computer's operations in a real-time program consists of "memory-reference instructions." Memory-reference instructions control the computer operations in taking data from memory and replacing it. These operations, along with register-manipulation operations, are actual machine operations that any computer sequences through. If we consider only word length, the implication could be made that a 12-bit word contains too few bits to describe a large enough set of memory-reference instructions. The argument proceeds that if four extra bits are added, many more basic computer instructions should become available. Each one of these extra basic instructions would be equal to a combination of the simpler instructions available on a 12-bit machine. We will see that this argument, good in theory, does not hold up for computers which are currently offered.

Figure 2 lists the basic memory-reference instructions of three representative computers. *Six memory-reference instructions* are available with the *12-bit* computer; *13 memory-reference instructions* in the 18-bit; and *14 memory-reference instructions* with the 16-bit model. Note that some 16-bit machines have as few as four or five memory-reference instructions. Let's review these lists in detail and consider the usability of the extra instructions which accompany longer word lengths.

MEMORY REFERENCE INSTRUCTION REVIEW

It is worth taking a moment to review the basic memory reference instructions to find out *what we are really talking about.* A computer operates by going through a list of instructions (stored program) that are sequentially decoded. The decoding of these instructions informs the hardware

*TM, VIDAR Corporation

logic of the computer to remove information from specific memory locations, manipulate the information, and return the information to memory. In some instances, the information is removed from memory locations, manipulated, and sent to the I/O devices. The memory-reference instructions describe how to manipulate this information and the sequence in which to proceed through the stored program. These instructions appear as 12, 16 or 18-bit binary words. Figure 1 shows the arrangement of groups of bits within these binary words. Figure 1 shows how the computer interprets these words as memory addresses, operation instructions, etc. Figure 2 is a listing of easily remembered mnemonics describing these memory-reference instructions. Figure 1 shows that there are eight memory-reference instructions possible with a 12-bit computer word. There are 16 possible memory-reference instructions with either 16- or 18-bit words (Figure 1). *Does substantial improvement accrue from the additional memory-reference instructions?*

The first instruction in each list is the logical "AND" operation between information contained in a specific memory location and information in the accumulator. In the 16/18-bit examples, an "Exclusive Or" operation is also possible while the 16-bit unit also has an "Inclusive Or." These three operations allow specific bit manipulation or "masking" of information that is in the accumulator or in a memory location. They are generally used in formatting information prior to sending it out of the computer. The next instruction on the list is the "TAD" instruction (Two's Complement Add). This specifies a *binary addition* of the information from the core memory to the accumulator. On the 16/18-bit machines, there are other addition operations. A different arithmetic system using a one's-complement add operation is available in the 18-bit machine (ADD). In addition, the 18-bit machine has an instruction which clears the accumulator prior to loading it from a given memory location (LAC). The 16-bit machine used in our examples has *two* accumulators. Thus, four instructions are used for loading and addition (ADA, ADB, LDA, LDB). Note that *it takes four of the 14 memory reference instructions to do basically two operations.* The next basic memory reference instruction is the *increment-and skip-on-zero* instruction (ISZ). This is a test instruction that is used to affect the sequence of programming steps.

All three of the machines considered have the same instruction. The next instruction is the *deposit-and-clear-accumulator instruction* (DCA in the 12-bit machine, STA and STB in the 16-bit machine and DAC in the 18-bit machine). With this instruction the contents of the accumulator are deposited in a specific memory location. In the 18-bit computer, there is an additional instruction which will deposit zeros into a specified memory location (DZM). The next instruction is the *jump-to-a-sub-routine* (JMS or JSB) instruction, necessary to modify the sequence in which the computer goes through a stored program. *This instruction is identical in all three computers.* Next is a *jump instruction* (JMP) which does the same operation as the previous instruction with one exception. The previous instruction, the JMS/JSB automatically allows for a return to the original instruction sequence, but the JMP instruction does not. The 18-bit machine offers an additional memory-reference instruction called the "XCT" instruction. This instruction allows you to execute one remote instruction out of the normal sequence. *This is a nicety but is not particularly necessary.*

The one memory-reference instruction that the

Figure 2

List of Memory Reference Instructions

TYPICAL 12-BIT COMPUTER	TYPICAL 16-BIT COMPUTER	TYPICAL 18-BIT COMPUTER
6 Memory Reference Instructions	14 Memory Reference Instructions	13 Memory Reference Instructions
Mnemonics	*Mnemonics*	*Mnemonics*
AND	AND, XOR, IOR	AND, XOR
TAD	ADA/B(2), LDA/B(2)	TAD, ADD, LAC
ISZ	ISZ	ISZ
DCA	STA/B(2)	DAC, DZM
JMS	JSB	JMS, CAL
JMP	JMP	JMP, XCT
	CPA/B(2)	SAD
1.5 MSEC CYCLE TIME	2.0 MSEC CYCLE TIME	1.0, 1.5 MSEC CYCLE TIME

12-bit computer does not have that is available in both of the larger computers is the *"skip-if-the-accumulator-differs" instruction (SAD),* also called the "compare-the-accumulator" instruction (CPA/B). This instruction compares the content of the accumulator (or accumulators) with a specific memory location. If there is a difference, the next instruction in the sequence is skipped. Of all the memory-reference instruction differences between the 12, 16, and 18-bit machines, *this is the only significantly different instruction* available on larger word length machines. This instruction saves from two to five steps each time it is used as compared to the 12-bit computer. But, not all 16-bit machines have this comparison-and-skip capability as a memory-reference instruction.

In summary, what extra instructions are available with a longer word length? Of the eight additional instructions available on the 16-bit computer, four are duplicates required because of the two accumulators. While two accumulators may be of value, their availability (and thus these extra instructions) resulted from overall hardware design; they are not a product of word length. Three of the remaining instructions are only small conveniences, while the "compare" instruction is a truly useful addition. On the 18-bit machine, six of seven extra commands are simply variations on the basic instructions, while the "skip-on-difference" instruction is a useful extra.

Evaluation of Memory-Reference Instructions

Speed: In applying the three evaluation criteria to the memory-reference instructions, what conclusions can be drawn? Memory-reference instructions typically take two machine-cycle times. Referring to Figure 2, it can be seen that the 12-bit computer chosen has a cycle time of 1.5μs. Therefore, memory-reference instructions take 3.0μs. The 18-bit computer has machine-cycle times of 1.0 to 1.5μs, depending upon the model. The high-speed 16-bit machine discussed here has a 2μs cycle time. The larger the repertoire of memory-reference instructions, the fewer steps normally required, and the smaller number of cycle times required to complete a given task.

Many manufacturers, when they invest in a larger memory, as required by a 16-bit word length, *use a slower memory to keep costs down.* This increases the machine-cycle time. An example of this exists between 1.5μs of the 12-bit unit and the 2μs of this 16-bit computer. The latter computer runs 25% slower on every memory-reference instruction. And note that this is not a "slow" computer by industry standards! Many of the 16-bit machines run two to four times slower than this. Thus, although the additional memory-reference instructions *will* eliminate some program steps (10 to 15% at most) the slower memory most often used in larger word length computers causes a net reduction in speed. Note, however, that the 18-bit machine chosen does have a fast memory, and so it will offer some speed advantage.

In considering the extra hardware accumulator of the 16-bit machine, it should be kept in mind that a computer operates in serial fashion down the program instruction list. The extra accumulator will allow some additional inter-register operations which result in an increase in speed in completing a given task. However, it halves the effective number of memory-reference instructions available with the 16-bit machine.

Memory Requirements: The next evaluation criterion that should be applied to the difference between 12 vs. 16-bit memory-reference instructions is the amount of core memory required by a given program. Because of its larger repertoire of memory-reference instructions, 16/18-bit machines should require fewer memory locations to write out a sequence of programming steps to accomplish a given task. Based on programs written in VIDAC, *it has been estimated that at worst case 10% of the 12-bit memory-reference instructions could be eliminated* using the fuller repertoire of 16/18-bit computers. This would be an overall reduction in the number of memory locations required to write a specific D-DAS program. Because a program occupies between 25 and 75% of core memory, *a saving of 2% to 7% of core memory would accrue by using* 16/18-bit computers. Although real, this saving is small. A much more significant factor is the memory-utilization efficiency of the software supplied with each machine.

Ease of Use: The third evaluation criterion we should consider in regard to memory-reference instructions is ease of use. For most users, the basic memory-reference instructions available on the 12-bit machine are actually about as many operations as can be remembered and used at any one time. Many users do not remember all the additional instructions available in the larger machines nor are they sufficiently concerned with program efficiency to use each in the best way. Full-time programmers would probably find these instructions useful. However, most small computers are programmed by system users who are not full-time programmers. For these users, especially, the ease of use of the software with a particular machine (does it have sub-routine calls?, pseudo-ops?, etc.) becomes far more important than word length.

SUMMARY

In comparing memory-reference instructions against the three criteria, we see that:

1 *There is a 10-15% speed loss in going from the 12-bit computer to the 16-bit computer, and a 10 to 15% gain in going to the 18-bit unit.* Thus, word length alone does not guarantee speed.

Also, another thing to watch for is that many "16-bit" machines are really 8-bit machines which use 2 cycle times to accomplish what we have assumed will

be done in one cycle time. Thus, even with the same cycle time they could be twice as slow as any of the machines we have discussed.

2 *Up to 7% more memory locations may be required in programming the 12-bit computer* as compared to some 16/18-bit computers.

3 In ease of use, there is no significant advantage in the 16/18-bit computers. The advantages of superior software (such as VIDAC) outweigh minor advantages in the basic memory-reference instructions. If ease of use is the major criteria, a potential user should consider a higher level language than the machine language we are discussing. This may require a 20-50% penalty in memory programming space requirements and in speed of operations in return for the facility of a FORTRAN or COBOL Compiler language. This would overshadow the other minor differences in memory-reference instructions.

MICROINSTRUCTIONS

The next type of computer instruction that we should consider are "microinstructions" or "register-to-register," "register operation," "register test instructions," etc. These are used to test data in the accumulator and make logical decisions about what operations the computer should do next. In addition, these instructions also shift the data within the accumulator, or shift information from one accumulator to another, or test the accumulator by deciding whether certain bits are one or zero. Such instructions are probably used more often in real-time computer programming than they are in the non-real-time data-processing type of programming.

There is a bit of fallacious logic that states that, because a 16-bit machine has four extra bits available to code additional microinstructions, 16 times as many internal register operations may be coded. The facts are that *in the 12-bit computer there are 41 available microinstructions.* In the *16-bit machine selected for comparison, there are 43 microinstructions.* In the 18-bit computer, there are approximately 50 microinstructions. There is absolutely no correlation between the number of bits in a computer microinstruction word and the number of microinstructions available. 40 to 50 microinstructions are as many microinstructions as a programmer can keep track of and a designer of a small computer can conceive of. Each microinstruction adds hardware. Thus, it is the manufacturer's balancing of cost versus capability that establishes the number of microinstructions, not word length. Presently, small computer manufacturers hold the number of microinstructions to 40 or 50.

IOT Microinstructions

Another item that bears directly on real-time computer D-DAS systems and is related directly to microinstructions are the IOT microinstructions. These program the input and output of data from the computer. They are generally treated as a register-type instruction because they control the transfer of data between the computer's main register (accumulator) and an outside device register. In the 12-bit and 18-bit machines, it is possible to send and receive information from *64 different external devices.* Six bits of the IOT instruction are set aside for addressing these 64 external registers. The remaining bits designate that it is an IOT instruction and allow the computer to check the status of the external device. In the *16-bit computer* used in this example, *six bits of the 16-bit word* are also set aside for addressing 64 external devices. However, the standard hardware that comes with this computer is capable of actually addressing or communicating with only 8 of these 64 devices. Optional hardware may be purchased to implement the full 64-device capability. Like the other microinstructions previously discussed, it is not the word length, but the hardware furnished that establishes the computer's input/output capabilities.

Review

It is possible to make computers with any number of microinstructions, but this is a matter of computer hardware and has nothing to do with the computer word length.

COMPUTER ARITHMETIC OPERATION

Much discussion of the relative merits of 12 vs. 16 vs. 18-bit word lengths centers around the arithmetic-number systems available. The following descriptions may be helpful.

Single Precision

To do "single precision" (single computer word) arithmetic with a 12-bit machine, there are only ±2,047 numbers available. To do single-precision arithmetic with a 16-bit machine, there are ±32,767 numbers available. To do single-precision arithmetic with an 18-bit computer, there are ±131,000 numbers available. See Figure 3.

Double Precision

If two words of core are used with every arithmetic operation (double precision), there are ±8.3 million numbers available with a 12-bit computer. With a 16-bit machine, a double-precision arithmetic number system contains ±2 billion numbers. With an 18-bit machine, a double-precision arithmetic system contains ±34 billion numbers. These sizes are derived by raising 2 to the power of the number of bits in a computer word, times the number of computer words assigned to each arithmetic operation, less one bit assigned as the plus-minus bit. See Figure 3.

A major factor affecting single- and double-precision arithmetic operations in a real-time computer system is the lack of a decimal point. It is possible to handle decimal points by multiplying any number containing a decimal point into a whole number. *This operation is known as scaling.* Thus, 15.5 becomes 155 and somewhere else in memory is stored the fact that this number has to be multiplied by 10^{-1} Scaling is an extremely tedious and difficult thing to do when arithmetic op-

erations are programmed. Scaling requires a larger memory allocation and substantially longer computer-operation times. It should be borne in mind that once a number is scaled, further scaling occurs with every arithmetic manipulation that involves that number.

It is obvious that there are very few operations of an arithmetic nature which could be performed in a single precision with a 12-bit computer. Plus or minus 2,000 is just too constricted a number system for the data accuracy that we normally associate with real-time D-DAS systems. People are not used to working with such a limited number system — consider that ±2,000 limits multiplication, for instance, to only 50 x 40! However, we can do some very basic D-DAS operations with a single-precision number system, such as: keeping track of channel ID, counting the number of scans, and limit checking. The same comments apply to 16/18-bit machines.

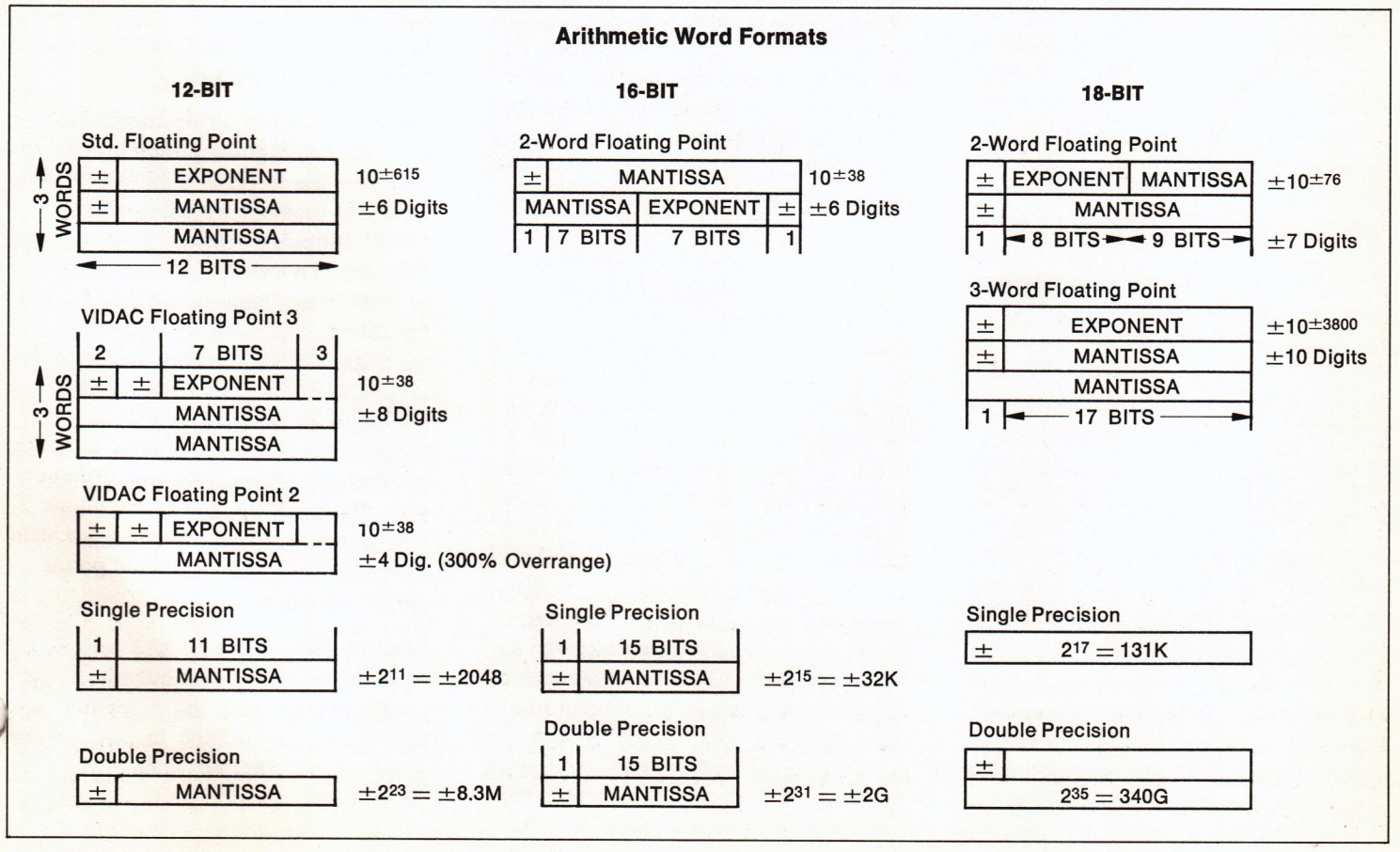
It is possible to check a positive or negative limit to one-tenth percent accuracy with a 12-bit single-precision comparison. With a 16-bit machine, it is practical to check a plus or minus limit to 0.005%. In most situations, users rarely know their limits to an accuracy of much better than 1%, and typically have limit accuracies of 10%. Thus, even the 12-bit machine with its single-precision accuracy of 0.1% is fine for almost all limit-checking requirements. *Thus, there is no inherent limitation in limit checking whether the computer uses 12, 16, or 18-bit words.*

Therefore, with the exception of limit comparisons, single-precision arithmetic with a 12, 16, or 18-bit computer is not a practical or reasonable consideration in D-DAS applications.

Floating Point

If single- or double-precision arithmetic operations are generally not used because they are too clumsy and cannot handle decimals, what is done? With VIDAR's VIDAC, we automatically use a floating-point number system which uses three 12-bit words for each floating-point number. A floating-point number system allows the user to have a specified number of digits multiplied by 10 to a specified power. Referring to Figure 3, we see that the VIDAC floating-point number system offers ±8 digits multiplied by $10^{\pm 38}$. This covers a staggering range of extremely small to extremely large numbers. The number of digits, and the power to which the base 10 can be raised in a floating-point number system, is directly related to the number of bits available to describe the numbers. An alternate three-word floating-point system (offered by the computer manu-

Figure 3



facturer) makes a slightly different selection of the division of the 36 available bits. Their number system has ±6 digits times $10^{\pm 615}$. Considering the VIDAR D-DAS data available, six digits are much more practical than eight. However, $10^{\pm 615}$ is not more practical than $10^{\pm 38}$. For many operations, a more practical floating-point system is the alternate VIDAC two-word floating-point system. This system describes a number of ±4 digits (with 300% overrange) times $10^{\pm 38}$. In other words, it has the same accuracy of digits as a reading from a 501/502 IDVM or a 520/521 IDVM with the 16-2/3ms measurement period, i.e., 0.01% resolution. The floating-point number system with the sample 16-bit computer is ±6 digits times $10^{\pm 38}$. The number system with the 18-bit computer is ±7 digits times $10^{\pm 76}$. See Figure 3.

Floating Point-Comparisons

The difference between 12, 16, or 18-bit computers is *not in the size* of the floating-point number system but rather in the *difference of accuracy* between four digits (with 300% overrange), six digits, or eight digits. If all of a user's information coming from an IDVM is four digits (with 300% overrange), then the VIDAC 12-bit two-word floating-point system is quite adequate. *If all* of a user's information coming from an IDVM is six digits, then the *16-bit two-word floating-point system* is a better choice. If all of a user's input information is eight digits, then the VIDAC three-word, eight digit, floating-point system is by far superior. In VIDAC, it is quite practical to alternate between the three-word, eight-digit, system and the two-word four-digit-plus-overrange system. There are no programming problems. Experience indicates that most of the data collected through the VIDAR D-DAS systems is the ±4 digit-plus-overrange accuracy.

Evaluation of Floating Point Systems

Speed: A two-word floating-point system is probably 30-40% faster than a three-word floating-point system. Therefore, *with six-digit information some advantage accrues with a 16-bit computer.* How often a user actually needs and uses six digits cannot be judged except to state that, based on VIDAR's past experience, we have found that about *80% of the measurements* taken are of the *four-digits (plus 300% overrange) accuracy.*

Memory Requirements: In terms of core memory required, it obviously requires 50% more word-memory locations to store tables and data with a three-word floating-point system. If a user is obtaining data that is ±4 digits (plus 300% overrange) accuracy, he should enter all his other parameters with the same order-of-magnitude accuracy. Accuracy is not enhanced by multiplying four-digit data with an eight-digit coefficient. It should be emphasized that in both data taking and in table storage, *VIDAC allows the programmer to select the accuracy of his floating-point number system.* VIDAC's floating-point systems do not straitjacket the programmer's core-storage allocation. It should also be remembered that a 4K computer D-DAS system uses a minimum of one-half of the core storage for the VIDAC "system module" and the application program. Therefore, generally, at best only one-half of the 4K memory is available for data and table storage. If it is assumed that 20% of the input data is of a greater accuracy than four digits (plus 300% overrange) and that one-half of the available core is used for data and parameter table storage (of equal accuracy), *then approximately 5% more core is required with a 12-bit computer* than would be required with a 16-bit computer. All these references to core-storage requirements assume that the 12-bit computer has a fixed number of 12-bit core-storage locations and the 16-bit computer has a fixed number of 16-bit core-storage locations. Many computers that are "16-bit" computers describe their core storage in terms of 8-bit "bytes," and, therefore, really contain only *one-half* as many 16-bit word core-storage locations as featured in their sales literature.

Ease of Use: If the computer manufacturer furnishes a *set of software* floating-point operations, there is no difference in the ease of use between a 12, 16, or 18-bit computer. However, many of the newer manufacturers of "16-bit computers" do not offer any floating-point software packages. As a result, in real-time data-acquisition problems, the *ease of use of these computers approaches zero on any absolute scale.* Therefore, the ease of use with a 16-bit computer can vary from equal to a 12-bit computer to totally unusable.

OVERALL CONCLUSIONS

Speed: It is possible to get a 10% to 15% speed improvement by using a 16 or 18-bit computer *if the three computers have equal cycle times.* Considering that available cycle times range from under 1 μsec to 8 μsec, cycle time is a much more important speed factor than word size.

Memory-Size Requirements: With certain types of high-accuracy input data, it is possible to require 10 to 20% less words of memory by using a 16-bit or 18-bit computer. However, if there are substantial needs for storage in excess of what is available in a basic 4K configuration, inexpensive mass storage, such as a disc memory or magnetic tape, should be considered. For example, increase in system price by adding a 32K disc is small as compared to the tremendous increase in available storage. The best solutions to storage requirements are usually obtained with computer peripherals . . . not by word-length selection.

Ease of Use Ease of use relates directly to the quality and quantity of software available with the computer system. It has no relation to the number of bits in a computer word.

VIDAR 77 Ortega Avenue, Mountain View, California 94040 • (415) 961-1000

Printed in U.S.A. 7/70